Демо-задача «Внутренний портал»

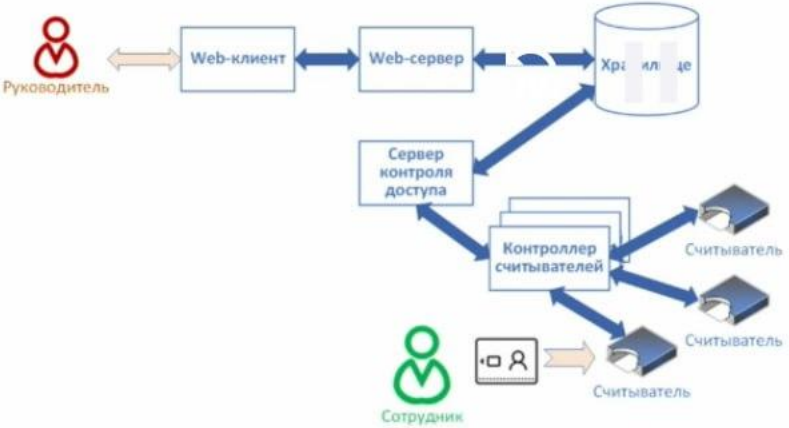
Дано:

- В компании требуется разработать клиент-серверное приложение для внутреннего использования, обеспечивающее автоматизацию контроля рабочего времени сотрудников
- Разработка системы будет вестись штатными программистами компании
- Рабочее время сотрудников фиксируется по индивидуальным проксимити-картам, которые сотрудники прикладывают к считывателю при входе и выходе из здания
- Доступ к журналу посещений и статистике рабочего времени сотрудников могут иметь только руководители

Что нужно сделать:

- Предложите вариант архитектуры такой системы с обоснованием каждой составной компоненты и набором технологий/инструментов для их реализации
- Предусмотрите возможность безотказной работы системы (доступность системы для прохода сотрудников должна составлять 0.999)
- Опишите API взаимодействия между компонентами системы, а также API взаимодействия с клиентом

Решение: функциональная структура (1/2)



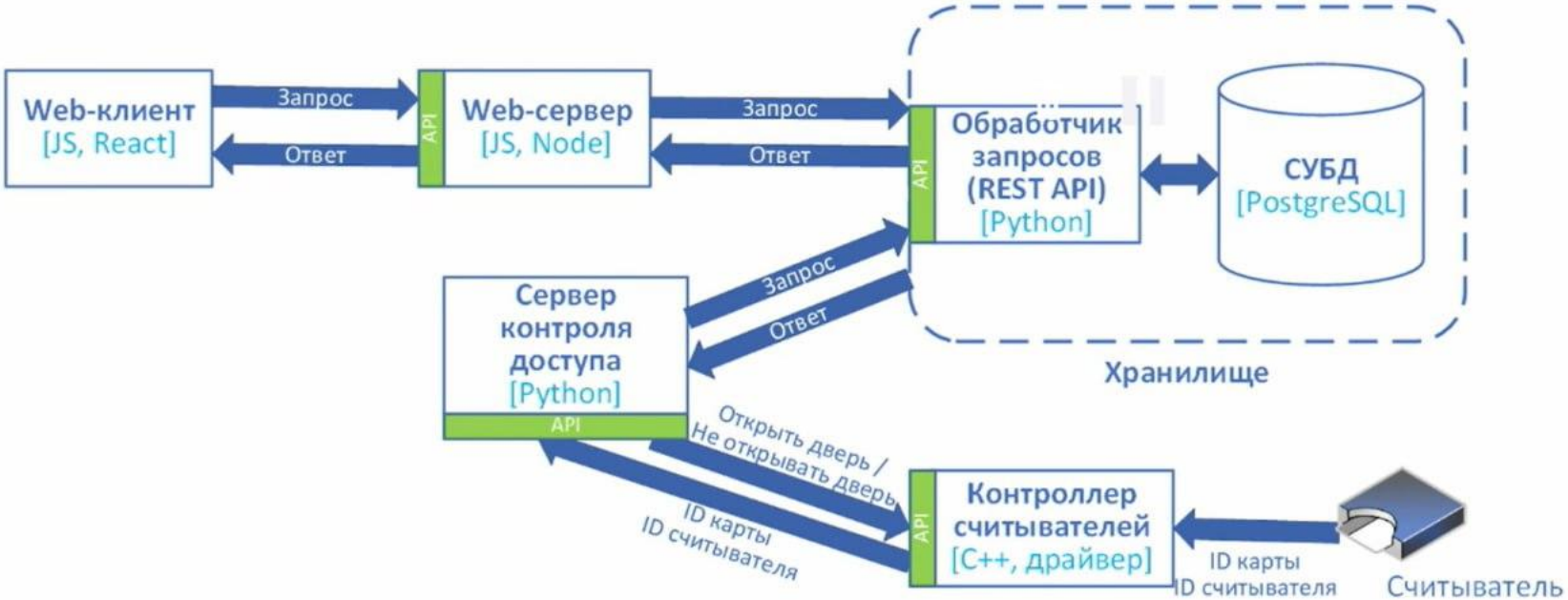
Решение: функциональная структура (2/2)

Функциональная структура – описание компонентов

- Web-клиент – взаимодействие с пользователем, получение выборки из хранилища по заданным фильтрам (пользователь, временной период, …)
- Web-сервер – проверка прав доступа к функциям web-клиента, обработка запросов и предоставление информации из хранилища
- Сервер контроля доступа – обработка событий «Считывание идентификатора карты» с последующим открытием привязанной к считывателю двери и фиксацией этого события в хранилище
- Контроллер считывателей – взаимодействие со считывателями и электронными замками
- Хранилище – хранение информации о пользователях, их правах доступа, картах доступа, считывателях, электронных замках, истории использования карт доступа

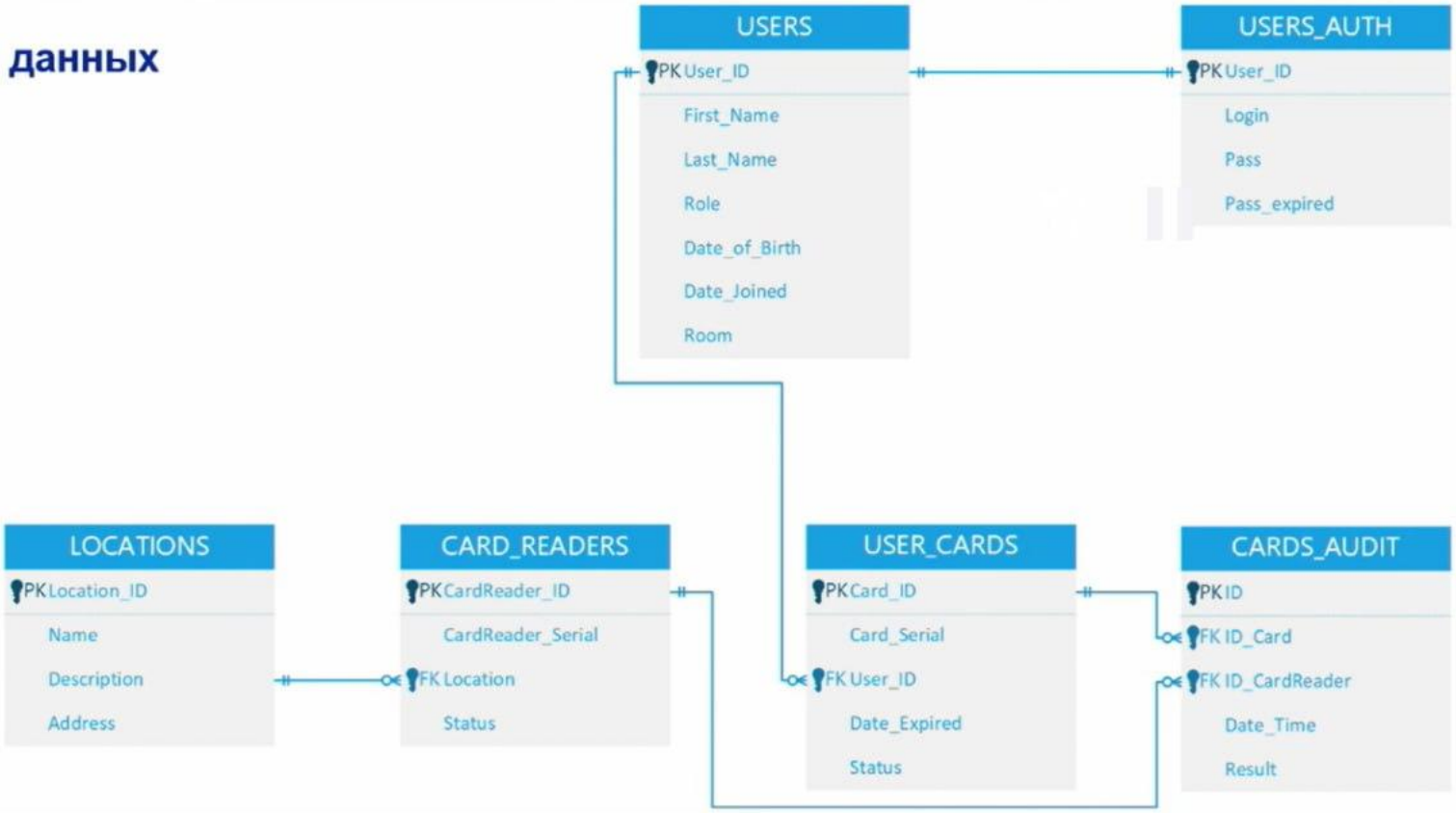
Решение: архитектура системы

Архитектура системы



Решение: архитектура данных (1/2)

Схема данных



Решение: архитектура данных (2/2)

Описание сущностей

- USERS – сотрудники компании
  (ФИО, дата принятия на работу, должность, кабинет)
- USERS_AUTH – информация для проверки прав доступа к функциям системы
  (логин, пароль, срок действия пароля)
- LOCATIONS – места, где располагаются сотрудники и считыватели карт
  (описание, местоположение)
- CARD_READERS – считыватели карт
  (серийный номер, положение, статус использования)
- USER_CARDS – карты доступа сотрудников
  (серийный номер, сотрудник, дата окончания срока действия, статус использования)
- CARDS_AUDIT – журнал использования карт доступа
  (номер считывателя, номер карты, временная метка, результат операции)

Решение: API взаимодействия с пользователем

API взаимодействия web-клиента с сервером

- `POST /auth` – аутентификация пользователя
- `GET /userRole/{userId}` – получение роли пользователя
- `GET|POST /userStat/{userId}?start={}&end={}` – получение статистики об использовании карты доступа сотрудником за указанный временной период и подсчет общего рабочего времени
- `GET /userCard/{userId}` – получение информации о карте доступа сотрудника
- `POST /userCard/{userId}` – изменение информации о карте доступа сотрудника

Решение: API взаимодействия компонентов (1/3)

API взаимодействия с сервером контроля доступа

- `check(card_id, reader_id)` – проверка на возможность открытия двери
  входные параметры: ID карты, ID считывателя
  результат: `TRUE, FALSE`
  внешние эффекты: нет

Решение: API взаимодействия компонентов (2/3)

API взаимодействия с хранилищем (1/2)

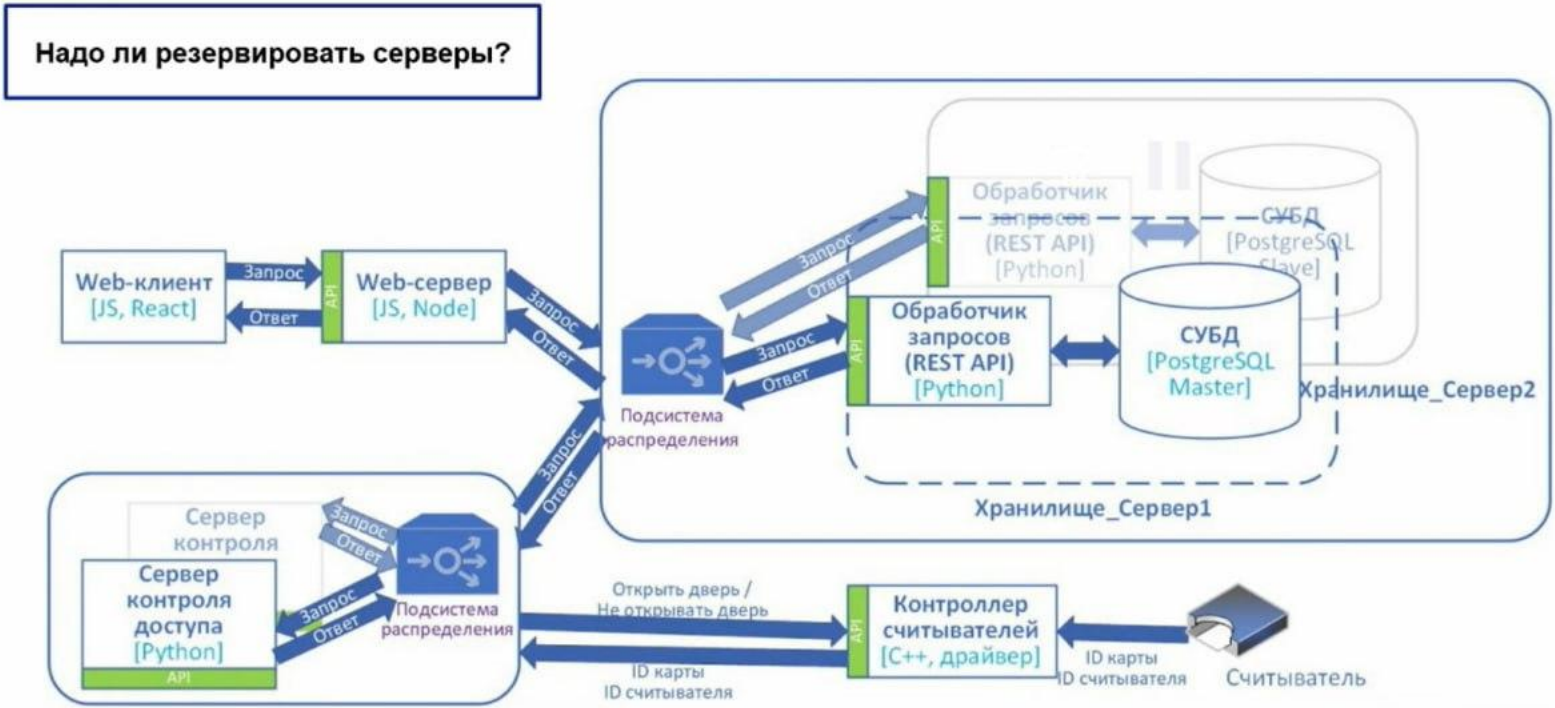
- `log(card_id, reader_id, result)` – добавление информации в таблицу CARDS_AUDIT об использовании карты доступа
  входные параметры: ID карты, ID считывателя, результат операции
  результат: `TRUE`, `FALSE`
  внешние эффекты: новая запись в таблице
- `userStat(userId, startDate, endDate)` – получение статистики об использовании карты доступа сотрудником за указанный период
  входные параметры: ID сотрудника, начало периода, конец периода
  результат: множество записей из таблицы CARDS_AUDIT
  внешние эффекты: нет
- `userCard(userId)` – получение информации по карте доступа сотрудника
  входные параметры: ID сотрудника
  результат: запись из таблицы USER_CARDS
  внешние эффекты: нет

Решение: API взаимодействия компонентов (3/3)

API взаимодействия с хранилищем (2/2)

- `userRole(userId)` – получение роли пользователя
  входные параметры: логин
  результат: `Role` из таблицы USERS
  внешние эффекты: нет
- `auth(login, pass)` – аутентификация пользователя
  входные параметры: логин, пароль
  результат: `TRUE`, `FALSE`
  внешние эффекты: нет

Решение: архитектура развертывания (1/2)



Решение: архитектура развертывания (2/2)

- клиентское приложение – тонкий клиент (браузер, оконное/мобильное приложение)
- web-сервер – web-сервер на основе Nginx с установленной серверной частью web-приложения, реализованной с использованием платформы NodeJS
- сервер контроля доступа – отдельная группа серверов, объединенных в локальную сеть вместе с балансировщиками, отвечающих на запросы от контроллера считывателей карт доступа
- хранилище – отдельная группа серверов с установленной СУБД PostgreSQL и настроенной репликацией
- контроллер считывателей – программа-драйвер (реализованная на языке программирования C++), взаимодействующая со считывателем карт доступа. Отправляет запрос на сервер контроля доступа, обрабатывает ответ и открывает электронный замок. Устанавливается на каждый контроллер (например, Arduino), подключенный к считывателю.